# Animationseditor: Anforderungen & Lösungen

Projekt: Animationswerkzeug zur
Visualisierung sozialer Handlungen

Projektbericht von
Stefan Schäfermeier

Dozenten:
Dr. Carola Eschenbach
Felix Lindner

Arbeitsbereich Wissens- und Sprachverarbeitung
des Fachbereichs Informatik
der Universität Hamburg

März 2011

# Inhaltsverzeichnis

# 1 Motivation

## 1.1 Mensch-Roboter-Interaktion

Im Forschungsbereich Mensch-Roboter-Interaktion wird analysiert, wie Roboter sich in Gegenwart von Menschen verhalten sollen. Insbesondere geht es darum, welches Verhalten von Robotern Menschen als nicht störend empfinden. Das „korrekte“ Verhalten hängt dabei auch von der jeweiligen Situation ab.

In einer Studie (Pacchierotti, Christensen & Jensfelt, 2005) wird beispielsweise die Mensch-Roboter-Interaktion auf einem engen Flur untersucht. Dabei geht es um die Fragestellung, wie Roboter einem Menschen auf einem engen Flur ausweichen sollen, mit dem Ziel, den Menschen so wenig wie möglich zu irritieren. In der Studie werden Parameter wie Geschwindigkeit und Abstand zum Menschen variiert und dann von den Versuchspersonen bewertet.

## 1.2 Heider-Simmel-Studie

Studien mit echten Robotern sind meist sehr aufwändig und teuer. Einfacher wäre es, solche Experimente auf einer abstrakten Ebene durchzuführen.

Grundlegend für solche Experimente ist eine Studie von Heider und Simmel (1944), in der Versuchspersonen Filme gezeigt wurden, in denen sich geometrische Figuren bewegen. Diese Bewegungen wurden als Aktionen von lebendigen Wesen interpretiert. Anhand der Art und Weise, wie die Bewegungen abgelaufen sind, wurde den geometrischen Figuren eine bestimmte Absicht unterstellt. Beispielsweise wurden schnelle Bewegungen als aggressiv und langsame Bewegungen als vorsichtig interpretiert. Figuren, die sich ausschließlich mit anderen Figuren bewegt haben, wurden als passiv angesehen. Nicht nur anhand der verschiedenen Bewegungen wurden den Figuren Eigenschaften zugeschrieben, sondern auch die Größe spielte eine Rolle. So wurde eine große Figur als stark/bedrohlich empfunden, während kleine Figuren eher schwach/hilfsbedürftig erschienen.

Nun ergeben sich die Fragen, ob man Experimente zum Raumverhalten von Robotern mittels solcher Filme durchführen kann und ob die Ergebnisse solcher Untersuchungen auf Experimente mit realen Robotern übertragbar sind. Um diese Fragen zu klären, müssen zuerst Versuche mit Animationssequenzen geometrischer Figuren durchgeführt werden.

# 2 Einleitung

## 2.1 Animationswerkzeug

Im Rahmen des Projekts „Animationswerkzeug zur Visualisierung sozialer Handlungen“ ist ein Werkzeug entstanden, mit dem geometrische Figuren und Animationen erstellt und abgespielt werden können. Es gibt zwar Programme, mit denen Heider-Simmel-Filme erstellt werden können, diese sind aber nicht speziell für diesen Zweck konstruiert worden. Sie bieten entweder mehr Funktionalität als benötigt und sind sehr komplex oder sie bieten nicht alle erwünschten Funktionalitäten.

Mit dem Animationswerkzeug können geometrische Figuren und Animationen einfach erstellt und der zeitliche Ablauf der Animationen verändert werden. Dadurch bietet das Animationswerkzeug die Möglichkeit, schnell und unkompliziert Heider-Simmel-Filme zu erstellen und mit diesen, Studien zum räumlichen Verhalten von Robotern durchzuführen.

Das Animationswerkzeug ist in vier Komponenten aufgeteilt:

1. Mit dem Grafikeditor können geometrische Figuren wie z.B. Rechtecke, Kreise, Polygone, Linien, usw. erstellt und platziert werden. Außerdem können Attribute wie die Größe und Farbe der Figur festgelegt werden.
2. Im Animationseditor werden den im Grafikeditor erstellten Figuren Animationen hinzugefügt. Der Benutzer kann zwischen verschiedenen Animationstypen wählen und das zeitliche Verhalten der Animationen bestimmen.
3. Der Player spielt die im Grafik- und Animationseditor erstellten Figuren und Animationen ab. Der Benutzer hat die Möglichkeit, die Animationssequenz zu pausieren und in ein Videoformat zu exportieren.
4. Das Modell enthält die geometrischen Formen und Animationen, die vom Grafik- bzw. Animationseditor erstellt werden.

## 2.2 Verwendete Bibliotheken

Im Animationswerkzeug werden geometrische Figuren und Animationen erstellt. Nun stellen sich die Fragen, wie das zugrunde liegende Modell aufgebaut sein soll und in was für einem Format diese Informationen persistiert werden.

### 2.2.1 Scalable Vector Graphics

Bei der Persistierung müssen alle Attribute der geometrischen Figuren und der Animationen gespeichert werden. Außerdem ist es sinnvoll, dass das Dateiformat eine zumindest ähnliche Struktur aufweist wie das Modell im Animationswerkzeug, um ein einfaches Laden und Speichern erstellter Animationssequenzen zu ermöglichen. All dies bietet SVG (W3C, 2010, Juni 22. http://www.w3.org/TR/SVG11/).

SVG ist ein mittlerweile weit verbreitetes XML-Format. In SVG gibt es eine Reihe an vordefinierten geometrischen Figuren und Attributen. Durch die Einbindung von SMIL (W3C, 2008, Dezember 1. http://www.w3.org/TR/SMIL/) können außerdem auch Animationen beschrieben werden. Die Elemente in SVG sind in einer hierarchischen Struktur angeordnet. Es gibt einen Wurzelknoten, dem geometrische Figuren hinzugefügt werden können. Die Figuren wiederum können mit verschieden Animationen versehen werden.

Da SVG ein XML-Format ist, können SVG-Dateien auch von Menschen gelesen werden. Dies bietet die Möglichkeit, Änderungen an einer Animationssequenz vorzunehmen, ohne das Animationswerkzeug dafür benutzen zu müssen.

Durch die weite Verbreitung des SVG-Formats können SVG-Dateien in diversen Internetbrowsern und Videoplayern abgespielt werden. Somit können Animationssequenzen, die mit dem Animationswerkzeug erstellt wurden, auch von Personen abgespielt werden, die dieses Werkzeug nicht besitzen.

### 2.2.2 Graphical Editing Framework

Sowohl im Animationseditor als auch im Grafikeditor müssen geometrische Figuren dargestellt werden und manipulierbar sein. Diese Funktionalitäten selbst zu schreiben, ist mit einem hohen Aufwand verbunden. Da es bereits Frameworks gibt, die die Anforderungen erfüllen, ist es sinnvoll, diese für das Animationswerkzeug zu verwenden.

Das Graphical Editing Framework (http://www.eclipse.org/gef/) ist ein Open Source Framework zum Erstellen grafischer Editoren. GEF basiert auf dem Framework Draw2D (http://www.eclipse.org/gef/draw2d/index.php), dessen Zweck es ist, geometrische Figuren darzustellen. GEF verbindet die Darstellung geometrischer Figuren mit dem darunterliegenden Modell und kümmert sich um Benutzereingaben.

Bei der Verbindung zwischen Modell und Darstellung sorgt GEF dafür, dass die hierarchische Struktur der Draw2D-Figuren der Struktur des Modells entspricht. Da auch SVG eine hierarchische Struktur besitzt, kann die hierarchische Struktur des Modells sowohl in der Darstellung als auch bei der Persistierung erhalten bleiben.

Weitere Details und die Anwendung von GEF sind im Projektbericht von Sven Röhling (Röhling, 2011) nachzulesen.

# 3 Anforderungen und Lösungen

Mit dem Animationseditor können Animationen erstellt, bearbeitet und gelöscht werden. Beim Erstellen einer Animation muss diese zuerst im Modell erzeugt und dann auf der grafischen Oberfläche dargestellt werden. Beim Bearbeiten von Animationen müssen die vom Benutzer geänderten Werte in das Modell geschrieben werden.
Im Gegensatz zum Grafikeditor, in dem geometrische Figuren erstellt werden, muss der Animationseditor dem Benutzer die Möglichkeit geben, den zeitlichen Ablauf von Animationen beschreiben zu können. Beim Grafikeditor werden die Attribute einer Figur einmal festgelegt und diese gelten dann über den gesamten Zeitraum. Mit dem Animationseditor ist es dagegen möglich, die Attributwerte der Figuren über die Zeit zu verändern. Dadurch werden an den Animationseditor andere Anforderungen als an den Grafikeditor gestellt.

## 3.1 Das Modell

### 3.1.1 Grundlegende Attribute

Im Modell werden alle Attribute einer Animation festgehalten. Dazu gehören zunächst die offensichtlichen Attribute Start- und Endzeitpunkt der Animation. Sind diese beiden Werte gegeben, so kann man leicht die Dauer einer Animation berechnen. Ein weiteres offensichtliches Attribut ist das Attribut einer geometrischen Figur, dessen Wert über die Zeit verändert (animiert) werden soll. Die geometrischen Figuren unterscheiden sich allerdings in der Anzahl und dem Typ der Attribute. Da der Zweck des Animationswerkzeugs das Erstellen Heider-Simmel-ähnlicher Filme ist, sollten die dort verwendeten Animation mit dem Animationseditor erstellbar sein.

Bei Heider-Simmel-Filmen können sich geometrische Figuren bewegen. Diese Bewegungen sind zum Teil sehr komplex. Es reicht also nicht, einer Figur die Möglichkeit zu bieten, sich in eine Richtung entlang einer Linie zu bewegen. Mit dem Animationseditor müssen Bewegungsanimationen erstellt werden können, in denen eine Figur auch Kurven entlang laufen kann. Hierfür können im Grafikeditor Pfade erstellt werden, die einer Bewegungsanimation hinzugefügt werden.
Weiterhin sind in Heider-Simmel-Filmen Rotationen zu beobachten, beispielsweise das Öffnen

einer Tür. Dabei werden Figuren nicht nur um den eigenen Mittelpunkt rotiert, sondern um einen beliebigen Punkt. Für eine Rotation muss also einerseits ein Rotationspunkt definiert werden, andererseits muss auch ein Winkel angegeben werden, der festlegt, um wie viel Grad die Figur rotiert werden soll.

Die Heider-Simmel-Filme sind in schwarz/weiß gehalten. Heutzutage hat man die Möglichkeit, solche Filme „farbenfroher“ zu gestalten. Dies wäre beispielsweise nützlich, wenn viele Figuren im Film enthalten sind, um sie besser unterscheiden zu können oder um mehrere Figuren einer Gruppe zuzuordnen oder auch um eine Figur besonders hervorzuheben. Durch den Einsatz verschiedener Farben kann auch die Animation eines Farbübergangs von Nutzen sein. Beispielsweise kann man so den Wechsel der Gruppenzugehörigkeit einer Figur beschreiben. Eine Farbanimation benötigt mindestens zwei Farben, um einen Farbübergang zu animieren.

Dem Benutzer stehen somit drei verschiedene Animationstypen zur Verfügung. Möglicherweise möchte der Benutzer aber ein Attribut animieren, das zu keinem der zuvor genannten Animationstypen passt. Bewegungsanimationen benötigen einen Pfad. Wenn der Benutzer eine Figur beispielsweise um 5 Pixel nach rechts bewegen möchte, wäre die Erstellung eines Pfades unnötig kompliziert. Die Animation des Radius eines Kreises ist mit den bisherigen Animationstypen überhaupt nicht möglich. Deshalb bietet der Animationseditor die Möglichkeit, eine Animation zu erstellen, bei der der Benutzer auswählen kann, welches der Attribute der geometrischen Figur animiert werden soll. Diese Attributanimation muss also nicht nur den Wert für ein Attribut speichern, sondern auch den Attributnamen.

### 3.1.2 Weitere Attribute

Animationen kennen ihren Start- und Endzeitpunkt und das zu animierende Attribut inklusive dessen Wertintervall(e). Im Folgenden geht es darum, wie eine Animation vom Startwert des zu animierenden Attributs zum Endwert gelangt.

**1. Berechnungsmodus**

SVG bietet bereits vier verschiedene Verfahren an, die für die Erstellung von Heider-Simmel-ähnlichen Filmen viele Möglichkeiten eröffnen:

1. Diskret

Die Animation beginnt mit dem Startwert und „springt“ am Ende der Animation auf den Endwert. Es findet also keine Interpolation statt, sondern der Startwert des Attributs wird mit dem Endwert überschrieben.

2. Lineare Interpolation

Die Werte zwischen Start- und Endwert werden interpoliert. Während der Animation wird der Startwert linear erhöht (oder verringert) bis am Ende der Animationsdauer der Endwert erreicht wird. Im Gegensatz zum diskreten Verfahren findet ein Übergang statt.

3. Gleichmäßiger Verlauf (SVG: paced)

Wenn die Animation mehr als zwei Werte enthält, führt die lineare Interpolation zu einem ungleichmäßigen Übergang, falls die Distanz zwischen den Werten nicht gleich ist.

Wenn beispielsweise die X-Koordinate mit den Werten 10, 20, 50 animiert werden soll, so startet die Animation mit dem Wert 10, erreicht nach der Hälfte der Animationsdauer den Wert 20 und am Ende den Wert 50. Da die Differenz beim ersten Intervall 10 und beim zweiten Intervall 30 beträgt, ist die Geschwindigkeit, mit der der Wert geändert wird, im ersten Intervall geringer als beim zweiten. Da die Geschwindigkeit pro Intervall konstant ist, wird die Geschwindigkeit beim Übergang von einem Intervall ins nächste sofort geändert. Eine solch plötzliche Geschwindigkeitsänderung sieht wenig natürlich aus. Beim genannten Beispiel könnte der Benutzer nur die Werte 10 und 50 wählen, um eine gleichmäßige Bewegung zu erhalten, aber wenn man den Wert 20 mit dem Wert 5 austauscht, wäre ein gleichmäßiger Verlauf nur durch zwei getrennte Animationen möglich. Im Falle eines „Zick-Zack-Kurses“ müsste man, je nach dessen Länge, sehr viele Animationen erstellen und die jeweils richtige Animationsdauer ausrechnen. Durch den gleichmäßigen Animationsverlauf wird eine konstante Geschwindigkeit während der gesamten Animationsdauer garantiert.

4. Nicht-lineare Interpolation (SVG: spline)

Bei der linearen Interpolation hat man innerhalb eines Intervalls eine konstante Geschwindigkeit. Für eine natürliche Bewegung benötigt man noch eine Möglichkeit, ein Beschleunigen bzw. Abbremsen zu beschreiben. In SVG wird hierfür eine kubische Bezierkurve verwendet. Damit kann die Geschwindigkeit innerhalb eines Intervalls variiert werden (Erläuterung siehe Kapitel 3.2.2.3 Geschwindigkeitsverlauf).

Eine Animation muss also Kenntnis darüber haben, welches der Verfahren verwendet werden soll. Im Gegensatz zu den ersten drei Verfahren müssen bei der nicht-linearen Interpolation zusätzliche Werte in der Animation gespeichert werden, nämlich die Daten der Bezierkurve, die den Geschwindigkeitsverlauf innerhalb eines Intervalls beschreibt. Bei mehreren Intervallen müssen entsprechend viele Daten gespeichert werden.

Eine kubische Bezierkurve ist durch vier Punkte definiert: Start-/Endpunkt und zwei Kontrollpunkte. Da Start- und Endpunkt der Kurve bereits durch den Start- und Endwert des Intervalls gegeben sind, reicht es aus, nur die Kontrollpunkte der Bezierkurve separat zu speichern.

**2. Dauer der Intervalle**

Der Benutzer kann nun den Geschwindigkeitsverlauf der Wertveränderung innerhalb eines Intervalls beeinflussen. Um die durchschnittliche Geschwindigkeit eines Intervalls zu verändern, muss der Benutzer die Dauer eines Intervalls ändern können. Dies wird erreicht, indem der Benutzer für jeden von ihm vorgegebenen Wert (Intervallgrenzen) bestimmen kann, wann dieser während der Animation erreicht werden soll. Beispielsweise kann der Benutzer bei einer Farbanimation mit drei Farben vorgeben, dass die zweite Farbe nach 60% der Animationsdauer erreicht werden soll.

Dabei sind allerdings einige Einschränkungen zu beachten:

1. Die relativen Zeitangaben müssen zwischen 0 und 1 liegen. Der zugehörige Wert kann nicht vor oder nach der Animation erreicht werden.
2. Der Startwert des ersten Intervalls wird immer zu Beginn einer Animation verwendet (also bei 0% der Animationsdauer)
3. Der Endwert des letzten Intervalls wird bei Interpolation immer am Ende einer Animation erreicht (bei 100% der Animationsdauer). Beim diskreten Berechnungsmodus wird keine Interpolation verwendet. Deshalb kann der letzte Wert vor Ende der Animation erreicht werden.
4. Die Zeitangaben müssen monoton steigend sein, da sich die Intervalle sonst überlappen könnten oder die Reihenfolge der Intervalle geändert werden könnte.
5. Beim Berechnungsmodus „paced“ (gleichmäßiger Verlauf) werden die Zeitangaben ignoriert, weil ansonsten keine konstante Geschwindigkeit während der gesamten Animation garantiert werden kann.

Das Verändern der Intervalldauer bietet also eine weitere Möglichkeit, den zeitlichen Ablauf einer Animation zu beeinflussen. Im Modell kann dies einfach berücksichtigt werden, indem die vom Benutzer angegebenen Zeitwerte als Attribut der Animation gehalten werden.

### 3.1.3 Die Architektur

Wie unter 3.1.1 erläutert und in Abbildung 3.1 zu sehen, gibt es 4 verschiedene Animationstypen. Da die Typen einige gemeinsame Attribute (Start- und Endpunkt, Attribute für das zeitliche Verhalten) haben, bietet es sich an, eine Klasse zu definieren, die alle gemeinsamen Attribute

enthält. Diese Klasse stellt allerdings keinen speziellen Animationstyp dar, deshalb sollte die Klasse abstrakt sein. Die Attribute, die die einzelnen Animationstypen voneinander unterscheiden, werden in separate Klasse ausgelagert. Somit wird jeder Animationstyp durch eine eigene Klasse repräsentiert. Alle Animationstyp-Klassen erben von der abstrakten Klasse. In Abbildung 3.1 ist die hierarchische Struktur des Modells als UML-Diagramm abgebildet.

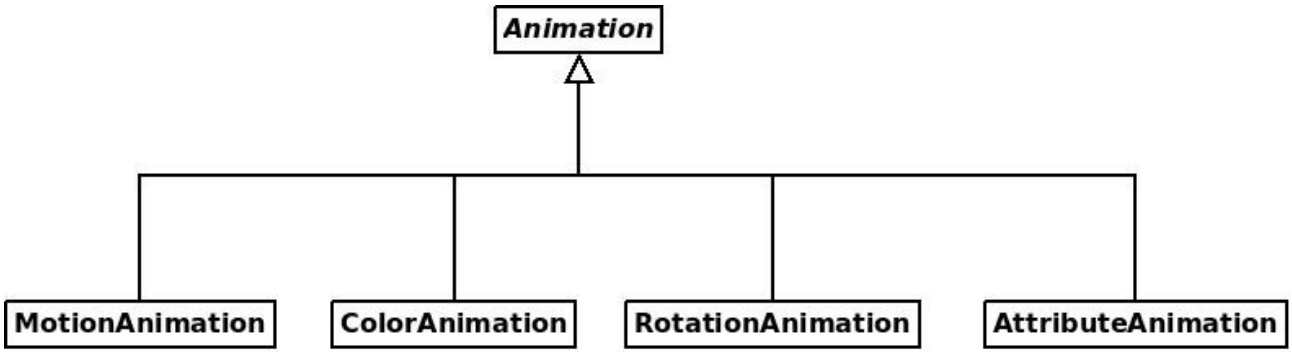


Abb. 3.1 UML-Diagramm der Modell-Architektur

### 3.2 Die grafische Oberfläche

Der Animationseditor ist ein Werkzeug zum Erstellen von Animationen, mit denen Attribute geometrischer Figuren über die Zeit verändert werden. Die Anforderungen können grundlegend in zwei Kategorien unterteilt werden:

1. Darstellung der gesamten Animationssequenz
2. Bereitstellung von Eingabemöglichkeiten für den Benutzer

#### 3.2.1 Darstellung

##### 3.2.1.1 Aufbau des Animationseditors

Animationen beziehen sich immer auf eine geometrische Figur bzw. auf ein Attribut einer Figur. Deshalb können Animationen nicht für sich alleine stehen. Diese Zuordnung zu einer Figur muss auch in der Darstellung berücksichtigt werden, damit der Benutzer klar erkennen kann, welche Animation zu welcher Figur gehört. Der Animationseditor muss also nicht nur die Animationen darstellen, sondern auch die geometrischen Figuren, die im Grafikeditor erstellt wurden. Da der Grafikeditor und der Animationseditor auf demselben Modell arbeiten, kann der Animationseditor vom Modell benachrichtigt werden, sobald eine neue geometrische Figur hinzugefügt oder entfernt wird. Im Gegensatz zum Grafikeditor spielt die Position der Figuren im Animationseditor keine Rolle, so dass man sie zur besseren Übersicht neben- oder untereinander auflisten kann.

Neben der Zuordnung zu einer Figur muss auch die zeitliche Abfolge der Animationen dargestellt werden. Für eine gute Übersicht bietet sich eine Zeitleiste an, wie man sie beispielsweise aus

Videobearbeitungsprogrammen oder auch TV-Zeitschriften kennt.

Bezüglich der Anordnung der Zeitachse und der Figuren bieten sich zwei Möglichkeiten an. Bei einer horizontalen Zeitachse und einer vertikalen Liste der Figuren, hat der Benutzer eine Übersicht darüber, welche Animationen zu einer Figur gehören und wie diese über die Zeit verteilt sind. Bei einer vertikalen Zeitachse kann der Benutzer besser erkennen, zu welchem Zeitpunkt welche Animationen aktiv sind, weil der Mensch es gewohnt ist, Dokumente horizontal zu lesen. Für den Animationseditor eignet sich eher die Variante mit einer horizontalen Zeitachse, da das zeitliche Verhalten der geometrischen Figuren und nicht die Darstellung einzelner Animationen im Vordergrund steht.

Die Liste der Figuren und die Zeitachse bilden somit eine Art zweidimensionales Koordinatensystem. Dadurch ist die Position einer Animation durch die Zuordnung zu einer Figur (Y-Koordinate) und dem Start- und Endpunkt der Animation (X-Koordinate) gegeben. Mit den X- und Y-Koordinaten sind bereits zwei Punkte vorhanden, die für die Darstellung einer Animation verwendet werden können. Da eine Linie zwischen den Punkten nur schwer zu erkennen ist, werden im Animationseditor Rechtecke für die Darstellung von Animation verwendet. Die Breite eines Rechtecks entspricht der Dauer der Animation.

In Abbildung 3.2 ist der Animationseditor mit vier geometrischer Figuren zu sehen, denen jeweils eine Animation hinzugefügt wurde. Die Animationen gehören unterschiedlichen Animationstypen an und haben verschiedene Startzeitpunkte.

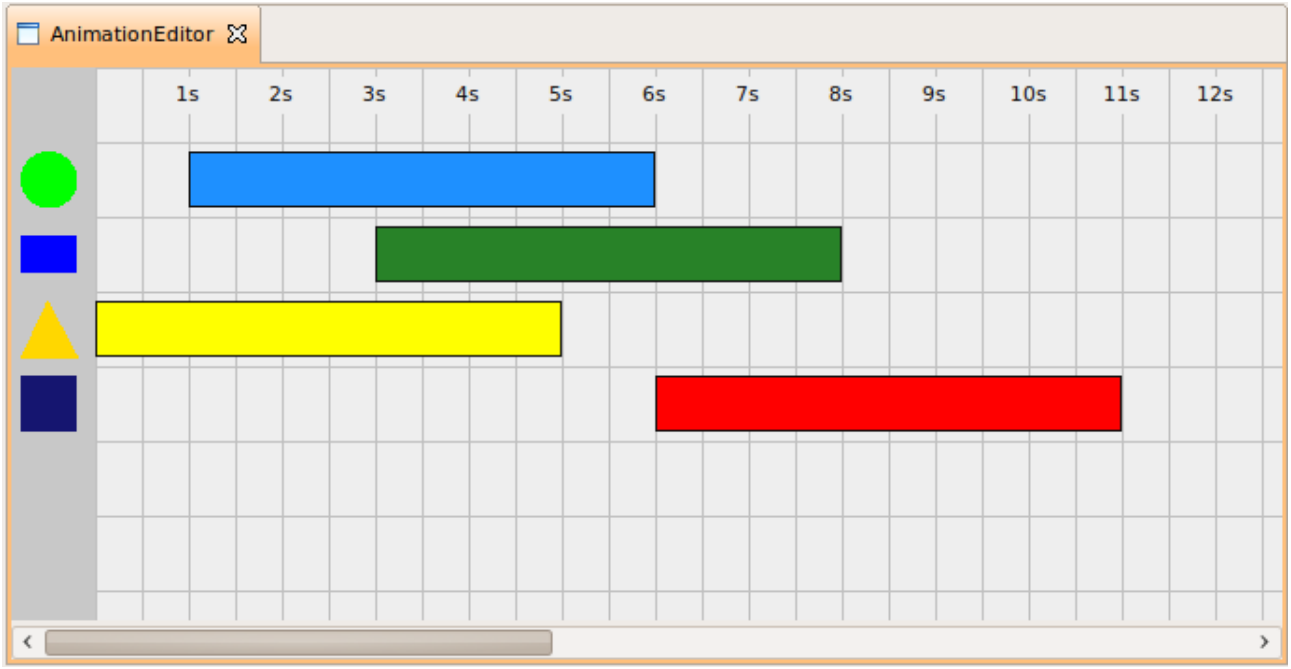


Abb. 3.2 Grafische Oberfläche des Animationseditors

Durch die Verwendung des Graphical Editing Frameworks (siehe Kapitel 2.2.2) erfolgt die Darstellung durch Draw2D. Da die hierarchische Struktur der Draw2D-Figuren der Struktur des Modells entspricht und die Animationen im Modell Kinder der geometrischen Figuren sind, ergibt sich folgende Struktur für die grafische Oberfläche des Animationseditors:
Die Zeitleiste und der Hintergrund (graue Spalte auf der linken Seite und Raster) bilden die Wurzelfigur. In der Hierarchieebene darunter befinden sich die einzelnen geometrischen Figuren. Die Rechtecke, die die Animationen repräsentieren, sind wiederum Kinder der geometrischen Figuren.
Da bei Draw2D Kindfiguren nur gezeichnet werden, wenn sie sich innerhalb ihrer Elternfiguren befinden, müssen die Draw2D-Figuren der geometrischen Figuren gesondert behandelt werden, weil ansonsten die Animationen nicht sichtbar sind. Um dieses Problem zu lösen, wird dem Wurzelknoten keine Draw2D-Figur hinzugefügt, die der entsprechenden geometrischen Figur entspricht, sondern es wird eine Draw2D-Figur erstellt, die einen transparenten Hintergrund hat und sich über die gesamte Breite des Animationseditors erstreckt. Diese Figur beinhaltet dann eine Draw2D-Figur, die zur entsprechenden geometrischen Figur passt. Eine geometrische Figur im Modell wird also durch diese „Zeilenfigur“ dargestellt. Animationen können nun in Form von Rechtecken einfach hinzugefügt werden.

#### 3.2.1.2 Weitere Aspekte der Darstellung

Wie im vorigen Kapitel erläutert, werden die geometrischen Figuren, die im Grafikeditor erstellt wurden, im Animationseditor in einer Liste dargestellt. Dabei muss sich nicht nur die Position der Figuren in der Darstellung von der im Modell unterscheiden. Auch die Größe muss angepasst werden, damit die Figuren nicht den Bereich der für sie vorgesehenen Spalte überschreiten. Die restlichen Eigenschaften wie Farbe und Form werden beibehalten, damit der Benutzer die Figuren im Animationseditor leichter den Figuren im Grafikeditor zuordnen kann.

Um die Benutzerfreundlichkeit weiter zu erhöhen, wird jeder Animationstyp in einer anderen Farbe dargestellt (siehe Abb. 3.2). Die Farbcodierung ist folgendermaßen:
- Attributanimation: Grün
- Farbanimation: Gelb
- Bewegungsanimation: Blau
- Rotationsanimation: Rot

Somit kann der Benutzer (nach einer Eingewöhnungsphase) auf den ersten Blick die Animationstypen voneinander unterscheiden, was wiederum zu einer besseren Übersicht führt.

Da die Dauer der gesamten Animationssequenz nicht limitiert ist, muss die Zeitleiste dynamisch erweiterbar sein. Platziert der Benutzer eine Animation über das Ende der Zeitleiste hinaus, so wird die Zeitleiste entsprechend erweitert. Wünschenswert wäre es, wenn sich die Zeitleiste bereits während des Verschiebens einer Animation aktualisiert. Da aber GEF die MouseEvents für Drag&Drop abfängt und behandelt, wäre der Aufwand sehr hoch, um diese Funktionalität realisieren zu können. Neben der Zeitleiste muss natürlich auch der Hintergrund (das Raster) dynamisch erweiterbar sein. Der Hintergrund muss allerdings nicht nur horizontal erweiterbar sein, sondern auch vertikal, da die Liste der geometrischen Figuren nicht begrenzt ist.
Eine weitere Funktion zur Erhöhung der Benutzerfreundlichkeit betrifft die Reihenfolge der geometrischen Figuren. Wird im Grafikeditor eine Figur erstellt, so wird ihr entsprechendes Pendant der Liste im Animationseditor unten angehängt. Sind viele Figuren in der Liste vorhanden, der Benutzer möchte aber nur wenige bearbeiten oder vergleichen, so wird dies schwierig wenn die Figuren über die gesamte Liste verteilt sind. Deshalb wird dem Benutzer die Möglichkeit gegeben, die Reihenfolge der geometrischen Figuren durch Drag&Drop zu ändern. Die zur Figur gehörenden Animationen müssen dabei mit verschoben werden. Da sich die geometrische Figur und die Animationen in derselben Draw2D-Figur (Zeilenfigur) befinden, kann die Reihenfolgeänderung durch das Verschieben der Zeilenfigur erreicht werden.

### 3.2.2 Eingabemöglichkeiten

Damit der Animationseditor Animationen anzeigen kann, müssen diese natürlich erst einmal vom Benutzer erstellt werden können. Außerdem muss der Animationseditor dem Benutzer die Möglichkeit geben, alle für eine Animation relevanten Eigenschaften verändern zu können.
Jede Animation ist an eine Figur gebunden. Deshalb muss der Benutzer im Animationseditor zuerst eine Figur auswählen, bevor er eine Animation erstellen kann. Aufgrund dieser Notwendigkeit bietet es sich an, das Erstellen von Animationen über ein Kontextmenü zu beginnen. So reicht ein Mausklick aus, um zu einer Liste der verschiedenen Animationstypen zu gelangen. Nach Auswahl eines Typs gelangt der Benutzer zu einem Dialogfeld, in dem die für den jeweiligen Animationstyp notwendigen Daten eingegeben werden können. Die einzige Ausnahme ist die Bewegungsanimation. Diese benötigt einen im Grafikeditor erstellten Pfad. Eine Auflistung aller Pfade ist schwierig, da für den Benutzer erkenntlich sein muss, welcher Eintrag in der Liste zu welchem Pfad gehört. Für den Benutzer ist es deutlich einfacher, einen Pfad im Grafikeditor zu selektieren, und dann eine Bewegungsanimation zu erstellen. Da Animationseditor und Grafikeditor voneinander getrennt sind, geht die Markierung des Pfades nicht verloren, wenn der Benutzer im Animationseditor das Kontextmenü aufruft. Allerdings muss der Animationseditor abfragen können,

ob im Grafikeditor ein Pfad selektiert ist. Da die Bewegungsanimation neben dem Pfad keine weiteren, speziell zum Animationstyp gehörenden Attribute hat, wird zum Erstellen der Animation kein Dialogfeld benötigt.
Bei den anderen Animationstypen muss der Benutzer Daten eingeben. Da diese sich je nach Animationstyp unterscheiden, wird beim Erstellen jeweils ein anderes Dialogfeld benötigt.
Das Kontextmenü bietet nicht nur die Möglichkeit, Animationen zu erstellen, sondern diese auch wieder zu löschen.

#### 3.2.2.1 Dialogfelder zur Erstellung von Animationen

1. Attributanimation:

Bei Attributanimationen kann ein beliebiges Attribut animiert werden. Deshalb müssen nicht nur Attributwerte festgelegt werden sondern auch das Attribut selbst. Welche Attribute vom Benutzer ausgewählt werden können, hängt davon ab, welcher geometrischen Figur die Animation hinzugefügt werden soll. Beispielsweise macht es wenig Sinn, bei einem Rechteck den (nicht vorhandenen) Radius verändern zu wollen. Pro Animation kann nur ein einziges Attribut ausgewählt werden, so dass es eine klare Zuordnung der Animation zum Attribut gibt. Dadurch kann der Benutzer die Animation eines einzelnen Attributs verschieben oder löschen, ohne die Animation anderer Attribute zu beeinflussen. Für eine Auswahl aus einer endlichen Menge eignet sich als grafisches Element eine Auswahlbox.
Für die Attributwerte werden ausschließlich Zahlen verwendet. Allerdings muss zwischen RGB-Farbwerten, die jeweils aus drei Zahlen im Bereich von 0 bis 255 bestehen, und den restlichen Attributen, die jeweils nur eine Zahl benötigen, unterschieden werden. Als grafische Elemente eignen sich Eingabefelder, die ausschließlich die Eingabe von Zahlen erlauben.
Da der Benutzer beliebig viele Attributwerte eingeben kann, werden alle Eingaben in einer Liste angezeigt, so dass der Benutzer einen Überblick über die bereits getätigten Eingaben hat und diese auch bearbeiten oder entfernen kann.
In Abbildung 3.3 ist das Dialogfeld abgebildet. Als Attribut ist die Y-Koordinate ausgewählt und der Liste wurden bereits zwei Werte hinzugefügt. Beim Klick auf den OK-Button wird eine Attributanimation erstellt, die die Y-Koordinate der zugeordneten geometrischen Figur von 20 auf 35 erhöht.

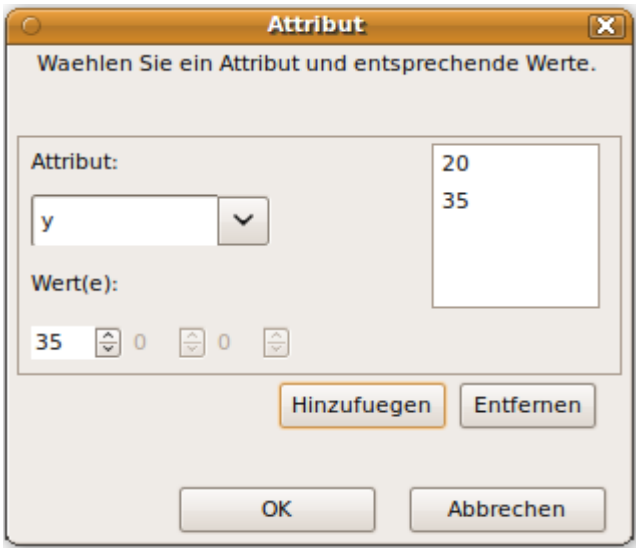


Abb. 3.3 Dialogfeld einer Attributanimation

2. Farbanimation:

Bei einer Farbanimation wird die Füllfarbe einer Figur animiert. Um eine solche Animation zu erstellen, muss der Benutzer die Möglichkeit haben, Farben auszuwählen. Die beste Variante zur Farbauswahl ist ein Farbdialog, bei dem der Benutzer nicht nur RGB-Werte eingeben kann, sondern eine Anzeige des gesamten Farbraums bekommt, in der eine Farbe ausgewählt werden kann. Eine Farbanimation kann allerdings mehrere Farbwerte enthalten, so dass ein einzelner Farbdialog nicht ausreicht. Deshalb wird noch ein weiteres Dialogfeld zur Verfügung gestellt, bei dem ein Farbdialog geöffnet werden kann, und das die ausgewählten Farben angezeigt. Das Dialogfeld bietet außerdem die Möglichkeit, die ausgewählten Farben wieder zu entfernen.

In Abbildung 3.4 sind beide Dialogfelder zu sehen. Der Farbdialog kann je nach Betriebssystem variieren. Beim Klick auf OK wird eine Farbanimation von Rot nach Blau erstellt.

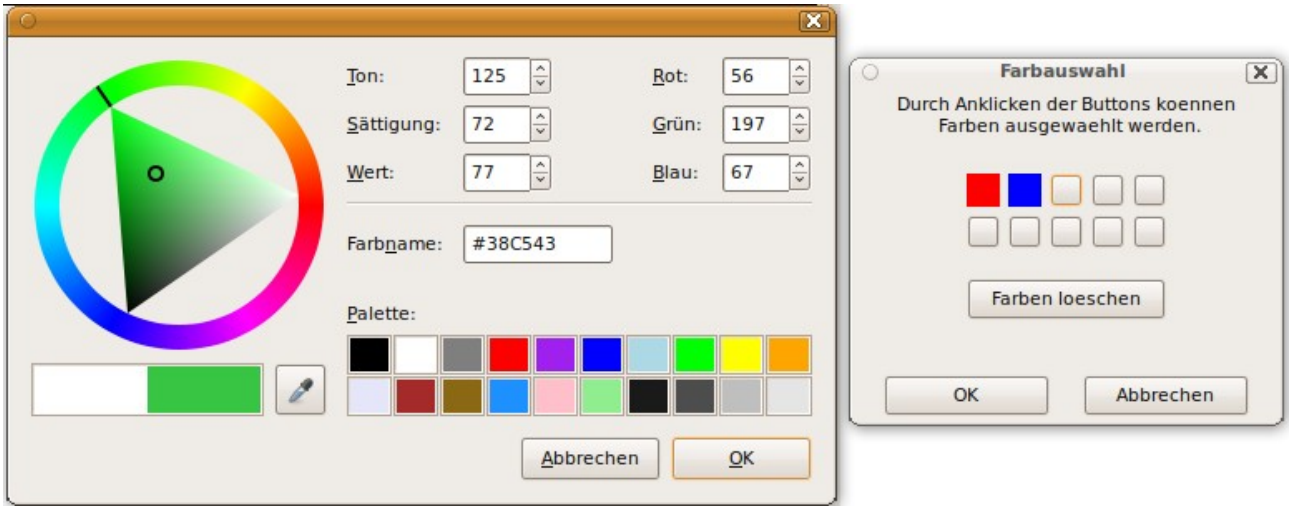


Abb. 3.4 Dialogfeld einer Farbanimation (rechts) und ein Farbdialog (links)

3. Rotationsanimation:

Eine Rotationsanimation benötigt einen Rotationspunkt, bestehend aus einer X- und Y-Koordinate, und einen Winkel. Insgesamt muss der Benutzer also drei Zahlenwerte eingeben können. Wie bereits beim Dialogfeld für die Attributanimation werden Eingabefelder verwendet, die nur die Eingabe von Zahlen erlauben. Die Eingaben des Benutzers werden zur Übersicht in einer Liste angezeigt und können auch wieder entfernt werden.

In Abbildung 3.5 ist das Dialogfeld zur Erstellung von Rotationsanimationen dargestellt. Die Eingabefelder sind untereinander angeordnet und auf der rechten Seite befindet sich die Liste.

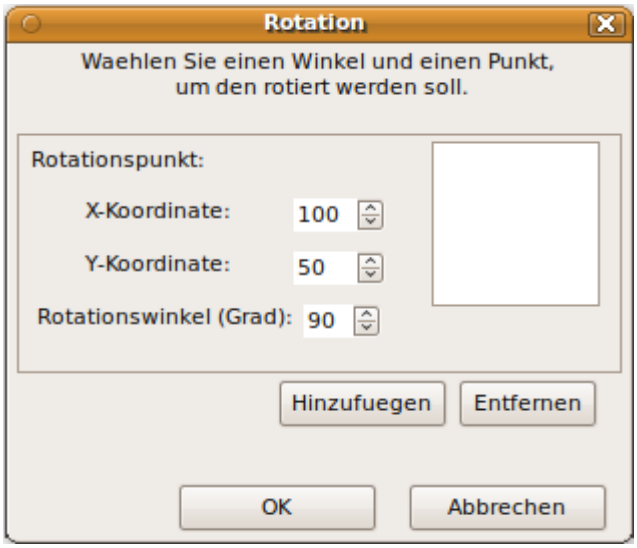


Abb. 3.5 Dialogfeld einer Rotationsanimation

**3.2.2.2 Bearbeiten von Animationen**

Beim Erstellen einer Animation muss der Benutzer je nach Animationstyp unterschiedliche Werte angeben. Eine Animation hat allerdings noch viele weitere Attribute, die während der Erstellung mit Standardwerten initialisiert werden. Diese müssen aber auch vom Benutzer bearbeitet werden können.

Start-, Endpunkt und Dauer:

Die wichtigsten Attribute sind der Zeitpunkt, bei dem die Animation beginnen soll, und die Dauer bzw. das Ende der Animation. Da diese Attribute womöglich häufiger angepasst werden, ist hierbei eine möglichst einfache Steuerung besonders wichtig. Da die X-Koordinate der Punkte auf der linken Seite des Rechtecks, das die Animation repräsentiert, anhand des Startpunkts der Animation berechnet wird, sollte auch eine Änderung dieser X-Koordinate dazu führen, dass der Startpunkt der Animation im Modell entsprechend geändert wird. Analog dazu kann der Endpunkt einer Animation verändert werden, indem die Punkte auf der rechten Seite des Rechtecks verschoben werden. Wird nur der Start- oder Endpunkt geändert, so ändert sich automatisch auch die Dauer der Animation.

Ein Verschieben der Animation, ohne dabei die Dauer zu ändern, kann per Drag&Drop des Rechtecks erreicht werden. Die Manipulation einer Rechteck-Figur wird bereits durch das Graphical Editing Framework bereitgestellt.

Weitere Attribute:
Durch ein Rechteck können Start-, Endpunkt und Dauer einer Animation grafisch dargestellt werden. Dies gilt allerdings nicht für die restlichen Attribute, da diese Strings als Werte haben oder aus einer Menge von Zahlen bestehen und vor allem keinen direkten Bezug zur Zeitleiste haben. Um die Übersichtlichkeit des Animationseditors nicht zu verringern, werden die restlichen Attribute und deren Werte in einem separaten Fenster, dem Eigenschaftseditor, aufgelistet. Dort können sie vom Benutzer bearbeitet werden. Neben diesen Attributen werden dort auch Start- und Endpunkt sowie die speziellen Attribute des jeweiligen Animationstyp angezeigt. Somit beinhaltet die Liste alle Attribute einer Animation. Da jede Animation eigene Werte hat, muss der Benutzer eine Animation auswählen, damit ihre Werte im separaten Fenster angezeigt werden. Weitere Informationen zum Eigenschaftseditor sind im Projektbericht von Tobias Staron (Staron, 2011) zu finden.
Die Attribute Fill, Start- und Endpunkt sowie der Berechnungsmodus können durch eine einfache Zahleneingabe oder über ein Dropdown-Menü verändert werden. Für die Werte der einzelnen Animationstypen können die Dialogfelder aufgerufen werden, die bereits beim Erstellen einer Animation verwendet werden (siehe Kapitel 3.2.2.1).

Die Attribute zur Beeinflussung des zeitlichen Ablaufs (Intervalldauer, Geschwindigkeitsverlauf siehe Kapitel 3.1.2) erfordern die Eingabe einer Zahlenkombination. Um das Bearbeiten dieser Attribute zu erleichtern, werden dem Benutzer Dialogfelder zur Verfügung gestellt.

#### 3.2.2.3 Dialogfelder zum Verändern des zeitlichen Ablaufs

1. Intervalldauer:
Um dem Benutzer das Ändern der Intervalldauer zu erleichtern, werden eine Zeitleiste, die die Dauer der gesamten Animation darstellt und die verfügbaren Animationswerte (Intervallgrenzen) angezeigt.
Mit diesen Informationen kann der Benutzer bestimmen, welcher Animationswert zu welchem Zeitpunkt während der Animation erreicht werden soll und damit wie lang die Intervalle sind.
Im Dialogfeld ist die Zeitleiste horizontal angeordnet und enthält relative Angaben bezüglich der Animationsdauer. Die Animationswerte werden durch Balken repräsentiert, welche den ihnen

zugeordneten Wert enthalten. Zur besseren Unterscheidung werden die Balken mit verschiedenen Farben dargestellt. Durch Verschieben der Balken entlang der Zeitleiste kann der Benutzer die Intervalldauer ändern.

In Abbildung 3.6 sind die Intervalle einer Attributanimation abgebildet, die die Breite einer Figur animiert. Die Balken zeigen an, dass der Wert 20 bei 30%, der Wert 30 bei 70% und der Wert 50 bei 100% der Animationsdauer erreicht werden soll.

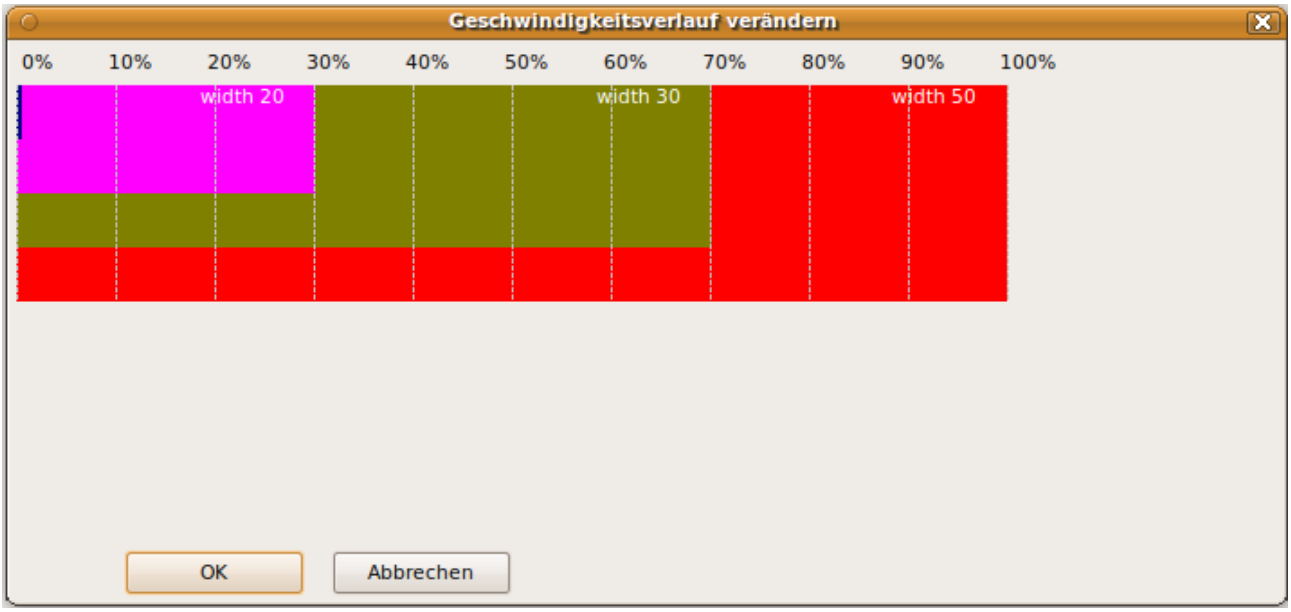


Abb. 3.6 Dialogfeld zur Manipulation der Intervalldauer

2. Geschwindigkeitsverlauf:

Durch Ändern der Intervalldauer kann der Benutzer die durchschnittliche Geschwindigkeit eines Intervalls beeinflussen. Allerdings kann die Geschwindigkeit innerhalb eines Intervalls nicht variiert werden, um beispielsweise ein „Abbremsen“ einer Figur zu ermöglichen. Hierfür bietet der Animationseditor ein weiteres Dialogfeld.

Da die Geschwindigkeit für jedes Intervall veränderbar ist, muss der Benutzer zuerst ein Intervall auswählen können. Wie im linken Dialogfeld der Abbildung 3.7 zu sehen ist, werden alle Intervalle aufgelistet und pro Intervall gibt es einen Button, mit dem das entsprechende Intervall ausgewählt werden kann. Außerdem gibt es pro Intervall ein Textfeld. Dort können Benutzer, die sich in SVG bereits auskennen, die Werte direkt eingeben. Als Standardwert wird eine konstante Geschwindigkeit verwendet. Durch Drücken eines der Buttons öffnet sich ein weiteres Dialogfeld, in dem dann der Geschwindigkeitsverlauf verändert werden kann (rechtes Dialogfeld in Abbildung 3.7).

Die Geschwindigkeit ist definiert als Strecke pro Zeit. Die Strecke ist in diesem Fall die

Wertveränderung vom Start- zum Endwert, also beispielsweise Farben bei einer Farbanimation. Die Zeit ist die Intervalldauer. Die Darstellung erfolgt in einem zweidimensionalen Koordinatensystem, bei dem die X-Achse die Zeit darstellt und die Y-Achse die Werte. Die Angabe, sowohl auf der X-Achse als auch auf der Y-Achse sind relativ (zwischen 0 und 1), so dass der Startpunkt des Intervalls der Punkt (0,0) ist und am Ende des Intervalls der Punkt (1,1) erreicht wird. Eine konstante Geschwindigkeit bedeutet, dass der Wert linear über die Zeit verändert wird. Dies wird durch eine Linie vom Punkt (0,0) zum Punkt (1,1) dargestellt. Eine einfache Linie kann allerdings nur verändert werden, indem mindestens einer ihrer Endpunkte verschoben wird. Da aber sowohl Startzeitpunkt und -wert als auch Endzeitpunkt und -wert fest sind, dürfen die Endpunkte nicht verschoben werden. Bei  einer kubischen Bezierkurve sind feste Endpunkte gegeben, der Benutzer hat aber trotzdem die Möglichkeit, durch Verschieben der Kontrollpunkte, die Geschwindigkeit (die Linie) zu verändern. Mit dem Koordinatensystem und der Bezierkurve kann der Benutzer gut erkennen, wie sich der Wert über die Zeit verändert.
In Abbildung 3.7 stellt die blaue Bezierkurve im rechten Dialogfeld den Geschwindigkeitsverlauf dar und die roten Rechtecke die Kontrollpunkte. Die Kurve gibt an, dass der Wert zu Beginn nur geringfügig geändert wird. Bis zur Mitte der Animationsdauer nimmt die Geschwindigkeit immer mehr zu. In der zweiten Hälfte der Animationsdauer nimmt die Geschwindigkeit, mit der der Wert verändert wird, wieder ab.

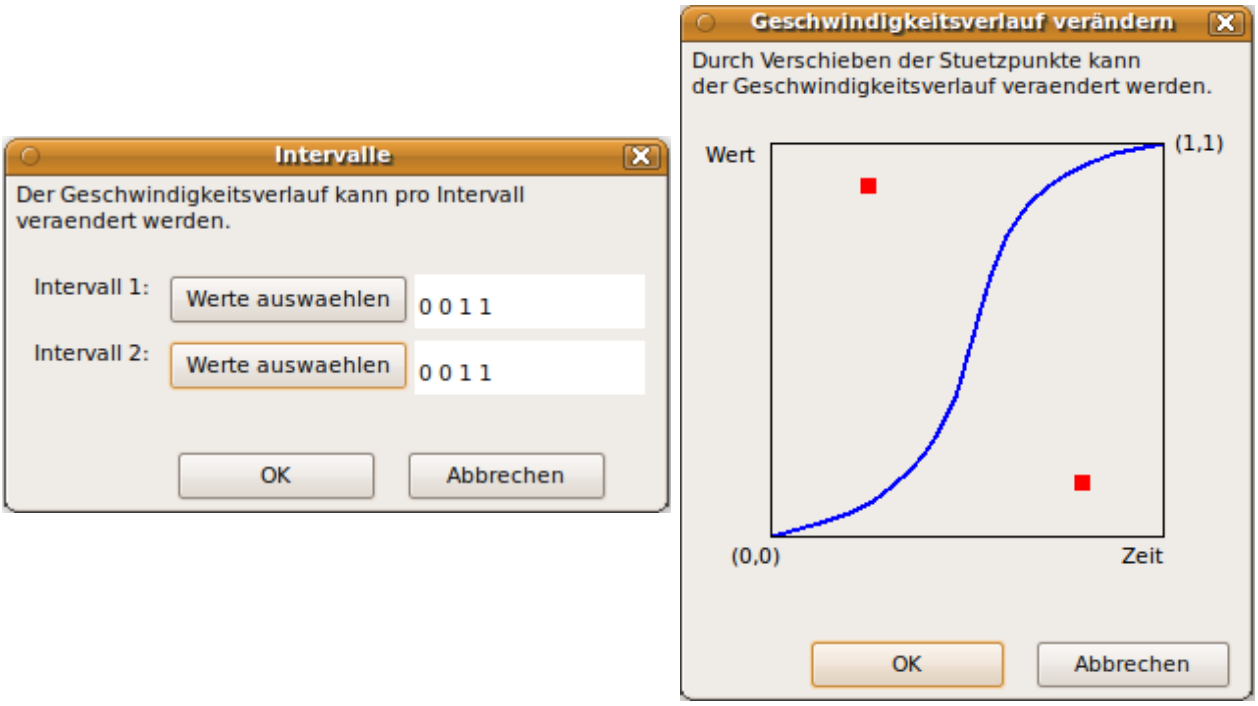


Abb. 3.7 Dialogfelder zur Auswahl eines Intervalls (links) und zum Ändern des Geschwindigkeitsverlaufs (rechts)

# 4 Evaluation

Der Zweck des Animationswerkzeugs ist das Erstellen Heider-Simmel-ähnlicher Filme, um damit Experimente zur Mensch-Roboter-Interaktion auf einer abstrakten Ebene nachzubilden. Mit dem Animationseditor sollen dafür Animationen erstellt werden können und zwar mit einem geringeren Aufwand, als wenn man dies direkt über eine SVG-Datei macht.

## 4.1 Schwierigkeiten bei Bewegungsanimationen

Mit dem Animationseditor lassen sich leicht Animationen erstellen. Die dafür benötigten Figuren werden automatisch im Animationseditor angezeigt, sobald sie im Grafikeditor erstellt wurden. Die Dialogfelder zum Erstellen eines bestimmten Animationstyps beinhalten einen kurzen Text, durch den der Benutzer erfährt, was in dem Dialogfeld eingestellt werden kann. Dies gilt allerdings nur für Attribut-, Farb- und Rotationsanimationen. Dass bei Bewegungsanimationen zuerst ein Pfad markiert werden muss, wird dem Benutzer verschwiegen.

Bei Bewegungsanimationen bestehen noch weitere Probleme. Die ihnen zugewiesenen Pfade können nicht ausgetauscht werden. Möchte der Benutzer den Pfad wechseln, muss er die Animation löschen und eine neue erstellen. Wird ein Pfad im Grafikeditor gelöscht, bleiben die zugehörigen Animationen weiterhin bestehen, so dass die Animationssequenz nicht mehr ausführbar ist. Sind viele Bewegungsanimationen vorhanden, so hat der Benutzer auch keine Möglichkeit, die Animationen manuell zu löschen, da nirgends angezeigt wird, welcher Pfad in welcher Animation steckt. War das Löschen des Pfades eine der letzten getätigten Operationen, könnte der Benutzer den Pfad per Undo wiederherstellen und durch Abspielen der Animationssequenz herausfinden, welche Animationen den Pfad enthalten. Ansonsten bliebe dem Benutzer nichts anders übrig, als die Animationssequenz als SVG-Datei zu speichern, diese zu ändern, so dass keine Animation mehr auf den gelöschten Pfad verweist, und die Datei wieder in das Animationswerkzeug zu laden. Da in der jetzigen Programmversion die Objekte einer geladenen SVG-Datei im Animationseditor nicht angezeigt werden, ist diese Lösung aber auch unbrauchbar. Der Benutzer wäre also gezwungen, alle Bewegungsanimation zu löschen, damit die Animationssequenz wieder ausführbar ist. Alle diese Varianten sind mit einem sehr hohen Aufwand verbunden.

## 4.2 Übersichtlichkeit des Animationseditors

Bei kleinen Animationssequenzen bietet der Animationseditor eine gute Übersicht über die Figuren und die Animationen, die dank der unterschiedlichen Farben leicht zu unterscheiden sind. Aufgrund einer fehlenden Zoom-Funktion kann die Übersicht bei großen Animationssequenzen nur über das

Vergrößern des Fenster erhalten bleiben. Die Möglichkeit, die Reihenfolge der Figuren zu ändern, kann die Übersicht wiederum verbessern, wenn der Benutzer nur an einer Teilmenge der Figuren arbeiten möchte. Problematisch sind allerdings Animationen, die zur selben Figur gehören und zur selben Zeit ablaufen. Die Animationen werden aufeinander gelegt, so dass eine oder mehrere Animationen verdeckt werden.

### 4.3 Realitätsnahe Bewegungen

In Heider-Simmel-Filmen sind sehr komplexe Bewegungen der Figuren zu sehen. Dabei „laufen" die Figuren nicht nur entlang eines Pfades, sondern sie ändern ihre Ausrichtung, wenn sie z.B. eine Kurve ablaufen. In SVG kann dies über ein Attribut in der Bewegungsanimation aktiviert werden. Der Animationseditor bietet in der aktuellen Version noch keine Möglichkeit, den Wert dieses Attributs festzulegen, so dass diese Funktionalität standardmäßig nicht aktiviert ist. Der Benutzer muss das Attribut also nachträglich in die SVG-Datei eintragen, was wiederum voraussetzt, dass SVG-Kenntnisse vorhanden sind.

### 4.4 Das Modell

1. Fill-Attribut:

Durch den Wert „Remove" beim Fill-Attribut springt der animierte Wert nach Ende der Animation auf den Startwert zurück. Speziell bei Bewegungsanimationen ist ein solches Verhalten in der Realität nicht möglich. Da mit dem Animationswerkzeug reale Experimente dargestellt werden sollen, müsste noch geklärt werden, ob es ausreicht, für das Fill-Attribut den Wert „Freeze" vorzugeben oder ob es Einsatzmöglichkeiten für den Wert „Remove" gibt.

2. Farbanimationen:

In SVG ist für die Animation von Farben ein separater Animationstyp definiert, obwohl Farbanimationen auch über eine Attributanimation (Attribut: fill oder stroke) möglich sind. Beim Erstellen Heider-Simmel-ähnlicher Filme haben Farbanimationen möglicherweise keine solch große Bedeutung, so dass ein separater Animationstyp vielleicht nicht gerechtfertigt ist.
Bei Farbanimationen wird standardmäßig die Füllfarbe verändert. Da im Modell nicht geprüft wird, zu welcher Figur die Farbanimation gehört, haben Farbanimationen keinen Effekt auf Linien, weil diese keine Füllfarbe besitzen.

3. Laden von Dateien:
Beim Laden von SVG-Dateien wird das Modell (noch) nicht aufgebaut, so dass der Animationseditor weder die Animationen noch die geometrischen Figuren aus der Datei anzeigen kann. Dadurch ist ein Bearbeiten geladener Datei nicht möglich.

### 4.5 Fazit

Mit dem Animationseditor können Animationen schneller und einfacher erstellt werden, als durch direktes Schreiben einer SVG-Datei und der Benutzer muss die Syntax von SVG nicht kennen. Allerdings können Benutzer, die keine Kenntnis von SVG haben, nicht-ausführbare SVG-Dokumente mit dem Animationseditor erstellen, so dass grundlegende SVG-Kenntnisse oder zumindest eine Anleitung zum Animationseditor nötig sind.
Durch die verschiedenen Animationstypen sind alle notwendigen Mittel gegeben, um Heider-Simmel-ähnliche Filme zu erstellen. Mithilfe des Grafikeditors und des Players können Animationssequenzen erstellt werden, ohne ein anderes Programm verwenden zu müssen.
Damit man den Animationseditor bzw. das gesamte Animationswerkzeug richtig einsetzen kann, muss aber zuvor das Laden von Dateien in den Animationseditor und den Grafikeditor implementiert werden. Ohne diese Funktionalität ist ein Erstellen komplexer Animationssequenzen deutlich erschwert, da die Animationssequenz bis zur Fertigstellung nicht aus dem Animationswerkzeug entfernt werden darf.

## Quellenverzeichnis

- Pacchierotti, E., Christensen, H. I., Jensfelt, P. (2005). Human-robot embodied interaction in hallway settings: a pilot user study. In *Proceedings of the IEEE International Workshop on Robots and Human Interactive Communication* (RO-MAN) pp. 164–171.

- Heider, F., Simmel, M. (1944). An experimental study of apparent behavior. *The American Journal of Psychology*, Vol. 57, No. 2, pp. 243-259, University of Illinois Press.

Projektberichte:
- Röhling, S. (2011). GEF als Werkzeug für Editoren. Bericht zum Projekt: Animationswerkzeug zur Visualisierung sozialer Handlungen, Universität Hamburg
- Staron, T. (2011). Eigenschaftseditor & Modellstruktur. Bericht zum Projekt: Animationswerkzeug zur Visualisierung sozialer Handlungen, Universität Hamburg

Spezifikationen:
- W3C, (2010, Juni 22). Scalable Vector Graphics (SVG) 1.1 (Second Edition). Besucht März 2, 2011, http://www.w3.org/TR/SVG11/

- W3C, (2008, Dezember 01). Synchronized Multimedia Integration Language (SMIL 3.0). Besucht März 4, 2011, http://www.w3.org/TR/SMIL/

Frameworks:
- Eclipse Foundation, Inc. Graphical Editing Framework. Besucht März 2, 2011, http://www.eclipse.org/gef/

- Eclipse Foundation, Inc. Draw2D. Besucht März 2, 2011, http://www.eclipse.org/gef/draw2d/index.php. Tutorial unter http://wiki.fernuni-hagen.de/eclipse/index.php/Draw2d